Applied Artificial Intelligence, 15:161–187, 2001

0883-9514/01 $12.00 +.00

LESSONS LEARNED DEPLOYING SPACE PLANNING SYSTEMS

STEVE CHIEN, TARA ESTLIN, FOREST FISHER, and GREG RABIDEAU
Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA

This article describes a number of lessons learned in deploying automated planning and scheduling systems for space applications at the Jet Propulsion Laboratory. Specifically, we describe issues relating to: how plan representation can affect plan quality; how to acquire, validate and maintain planning knowledge bases; and how planning and execution can be integrated. These issues are described in the context of several fielded systems in the areas of science data analysis, ground station automation, and spacecraft autonomy.

In the past several years, the artificial intelligence group at the Jet Propulsion Laboratory (JPL) has deployed a number of real-world planning systems, (Chien et al., 1997a). These systems were built to provide automated planning capability for a variety of domains including spacecraft command generation and validation, science data analysis, and control of deep space antennas. In fielding these systems, a number of unforeseen challenges were encountered, and some valuable insights into the process of building and deploying AI planning and scheduling systems have been gained. This paper describes some of the difficult issues encountered and discusses a sampling of the methods used to address them. By describing some of these issues, it is hoped that further interest in research in these areas is stimulated and close ties between the research, applied research, and application communities is fostered.

Specifically, in this paper, four key issues are focused on relating to planning and scheduling systems:

1. Integrating hierarchical task networks (HTNs) and operator-based planning systems (Estlin et al., 1997)—it has been found that most problems

Address correspondence to Steven Chien, Automated Planning & Scheduling, Jet Propulsion Laboratory, MS 126-347, Pasadena, CA 91109-8099. E-mail: steve.chien@jpl.nasa.gov

involve both parts that are most naturally represented in HTN format and parts that are most naturally represented in an operator-based format. Significant work is needed to formalize practical HTN and operator representations. In addition, the field needs to develop standard metaphors for representing commonly occurring planning domain patterns—similar to design patterns in software engineering (Pree, 1995).

2. Representing and reasoning about plan quality (Chien et al., 1996)—most planning and scheduling problems involve a wide range of solutions. Allowing the user to easily represent preferences over this space and constructing a planning system that can reason intelligently about this space is the key to correctly solving most planning problems. Most systems to date have used implicit or crude, approximate models of user objective functions.
3. Knowledge acquisition; representing knowledge and maintenance of planning knowledge bases (Smith et al., 1997; Chien, 1998)—researchers have not devoted sufficient effort to the intricacies of encoding, validating, and maintaining knowledge within a planning system. Our experiences were that this effort is a considerable fraction (if not the majority) of the cost of deployment and operations. Further work directed at the development of tools, processes, and methodologies to reduce this cost is a high payoff area of work for planning researchers and practitioners alike.
4. Integration of planning and execution (Chien et al., 1999); integration of the planning system into the operational context (Chien et al., 1996)—most planning research has focused on a batch formulation of the planning or scheduling problem. However, in reality, a planning system must be integrated into a process workflow, which rarely presents problems in such a clean fashion. Instead, plans are rarely generated from scratch, and are constantly revised and updated to reflect a changing understanding of the problem. In order to facilitate insertion of planning and scheduling technology into such an environment, planning and scheduling systems must provide the capabilities to work in concert with human processes and operate in continuous planning and plan revision modes.

It has been our experience that fielding applications provide excellent insights into research problems, some of which are not addressed by the academic research community. In many cases it is only through the fielding of real world systems that many shortcomings of current thinking become apparent.

The remainder of this paper is organized as follows. First, several systems are described that the artificial intelligence group has fielded in order to provide a context for describing the lessons learned. Next, each of the issues that have been briefly outlined above are described in further detail: integration of HTN and operator-based planning paradigms, representing and

reasoning about plan quality, knowledge base management, and integration of planning and execution. Finally, the conclusions of the paper are presented.

FIELDED SYSTEMS AND DOMAINS

To provide a context for this discussion, applications from three application areas are described, science data analysis antenna ground station automation, and spacecraft commanding. Within these three domains, several fielded AI planning and scheduling systems will be discussed. These are the application areas and systems that will be used later in the paper to highlight issues in developing and deploying planning systems.

Science Data Analysis

The artificial intelligence group has developed a number of planning systems to assist scientists in preparing and analyzing NASAs vast storehouse of scientific data. The first of these was the multimission VICAR planner (MVP), an automated image processing system. The multi-mission VICAR planner was first deployed in 1994 (Chien, 1994) and continued to evolve through 1996 (Chien & Mortensen, 1996). The successful MVP application lead to the later deployment of the automated SAR image processor (ASIP) system (Fisher et al., 1997; Fisher et al., 1998b). Both MVP and ASIP were designed to relive scientists from the significant effort of preparing science image data for analysis (Chien et al., 1997b). Previous to the existence of MVP and ASIP, a scientist or science team would employ programmers to link together processing libraries to prepare the science data. Both MVP and ASIP embody a significant body of knowledge regarding image processing and their respective science disciplines. This knowledge allows MVP and ASIP to accept a set of high level goals regarding the attributes of the desired image product and to produce an executable script to produce the requested processed data from the available raw data. In summary, these planning systems act as an intelligent interface agent to the science data—enabling scientists to request the data by specifying *what data they want* rather than being forced to know *how to produce it.*

In order to illustrate how MVP assists in VICAR planetary image processing, a typical example of MVP usage is provided to ground the problem and the inputs and outputs required by MVP. The three images, shown on the left in Figure 1 are of the planet Earth taken during the Galileo Earth 2

FIGURE 1. Raw and processed image files.

flyby in December 1992. However, many corrections and processing steps must be applied before the images can be used. First, errors in the compression and transmission of the data from the Galileo spacecraft to receivers on Earth have resulted in missing and noisy lines in the images. Line fill-in and spike removals are therefore desirable. Second, the images should be map-projected to correct for the spatial distortion that occurs when a spherical body is represented on a flat surface. Third, in order to combine the images, one needs to compute common points between the images and overlay them appropriately. Fourth, because multiple images taken with different camera states are being combined, the images should be radiometrically corrected before combination.

The multi-mission VICAR planner enables the user to input image processing goals through a graphical user interface with most goals as toggle buttons on the interface. A few options require entering some text—usually function parameters that will be included as literals in the appropriate place in the generated VICAR script. Figure 2 shows the processing goals input to MVP.

Using the image processing goals and its model of image processing procedures, MVP constructs a plan of image processing steps to achieve the

display automatic nav residual error
radiometric correction
missing line fillin
no limbs present in images
display automatic nav residual error
display man nav residual error
mosaic images
perform manual navigation
pixel spike removal
uneven bit weight correction
perform automatic navigation
perform manual navigation
map project with parameters ...
smooth mosaic seams using DN

FIGURE 2. Example problem goals.

requested goal. Figure 3 shows the plan structure for a portion of the overall image processing plan.

In this graph, nodes represent image processing actions (programs) and required image states to achieve the larger image processing goal. This plan is translated into a VICAR script which, when run, performs the desired image corrections and constructs a mosaicked image of the three input files. Figure 4 shows the MVP-generated VICAR code corresponding to this subplan which performs image navigation. Image navigation is the process of determining the matrix transformation to map from the 2-dimensional (line, sample) coordinate space of an image to a 3-dimensional coordinate space using information on the relative position of the imaging device (spacecraft position) and a model of the target being imaged (e.g., the planetary body) for a Galileo image. The finished result of the image processing task is shown at the right in Figure 1. The three original images now appear as a single mosaicked image map projected with missing and corrupted lines filled in.

Thus MVP allows the user to go directly from high-level image processing goals to an executable image processing program (called a procedure

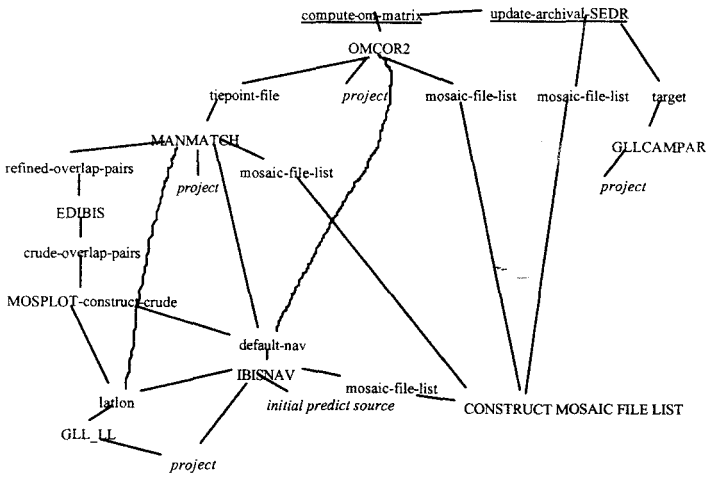

FIGURE 3. Subgoal graph for manual relative navigation of Galileo image files.

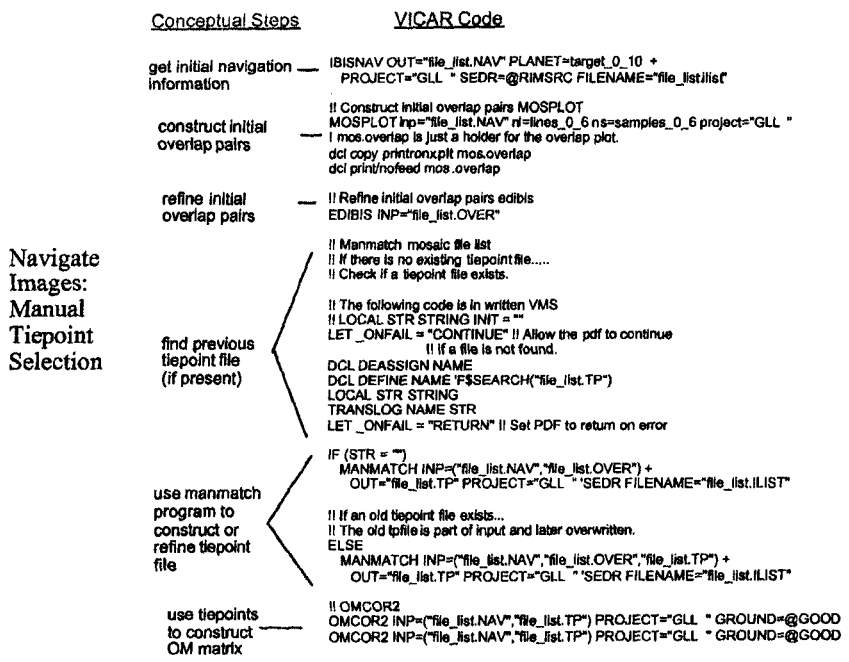

FIGURE 4. Sample VICAR code fragment.

definition file (PDF)). By insulating the user from many of the details of image processing, productivity is enhanced. The user can consider more directly the processing goals relevant to the end science analysis of the image, rather than being bogged down in the details such as file format, normalizing images, etc.

The multi-mission VICAR planner is often operated in a mixed-initiative planning mode where the user can help direct the planning process. In typical usage, the analyst receives a request, determines which goals are required to fill the request, and runs MVP to generate a VICAR script. The analysts then runs this script and visually inspects the produced image(s) to verify that the script has properly satisfied the request. In most cases, upon inspection, the analyst determines that some parameters need to be modified subjectively or goals reconsidered in context. This process typically continues several iterations until the analyst is satisfied with the image product.

Analysts estimated that MVP reduces the effort to generate an initial PDF for an expert analyst from 1/2 a day to 15 minutes and reduces the effort for a novice analyst from several days to one hour, representing over an order of magnitude in speedup. The analysts also judged that the quality of the PDFs produced using MVP are comparable to the quality of completely manually derived PDFs.

Ground Station Automation for the Deep Space Network (DSN)

In this subsection the NASA domain for deep space network (DSN) ground station automation is described. The DSN was established in 1958 and has since evolved into the largest and most sensitive scientific telecommunications and radio navigation network in the world. The purpose of the DSN is to support unmanned interplanetary spacecraft missions and radio and radar astronomy observations taken in the exploration of space. There are three deep space communications complexes located in Camberra, Australia, Madrid, Spain, and Goldstone, California. Each DSN complex operates a set of deep space stations consisting of a 70-meter, 34-meter, and 26-meter antenna. The function of the DSN is to receive telemetry signals from spacecraft, transmit commands that control spacecraft operating modes, generate the radio navigation data used to locate and guide a spacecraft to its destination, and acquire flight radio science, radio, and radar astronomy, very long baseline interferometry (VLBI), and geodynamics measurements.

In the past, the process of operating such stations has been labor and knowledge intensive. Recently, efforts have been made to reduce the cost of operations. One such effort has been in the area of automation. Many approaches have been applied to automation control/commanding of different types of systems. The AI group at JPL have worked on automating the generation of control/command sequences which can be run as control scripts to operate the station (Fisher et al., 1999).

The deep space terminal (DS-T) is a prototype 34-meter deep space communications station which was developed as a technology demonstration of *fully autonomous, lights-out*, operations (Fisher et al., 1998a). Figure 5 shows the 34-meter complex that was automated during the 34-meter demonstration. In the DS-T concept, each DS-T station operates autonomously, performing tracks in a largely independent fashion. When requested to perform

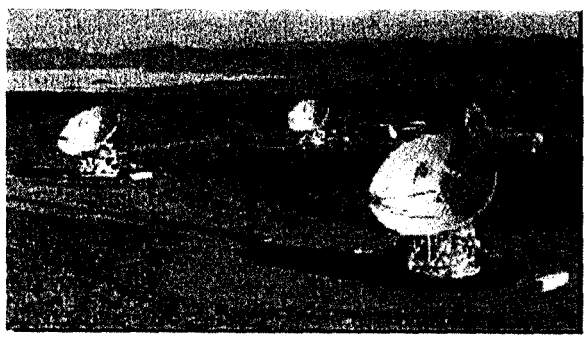

FIGURE 5. 34m BWG antennas at Goldstone.

a track, the DS-T station performs a number of tasks (at appropriate times) required to execute the track. First, the DS-T station uses appropriate spacecraft navigation ephemeris and predict generation software in order to produce the necessary antenna and receiver predict information required to perform the track. Next, the DS-T station executes the precalibration process, in which the antenna and appropriate subsystems (e.g., receiver, exciter, telemetry processor, etc.) are configured in anticipation of the track. During the actual track, the signal from the spacecraft must be acquired and the antenna and subsystems must be commanded to retain the signal, adjust for changes in the signal (such as changes in bit rate or modulation index as transmitted by the spacecraft), and perform error recovery. Finally, at the completion of the track, the station must be returned to an appropriate standby state in preparation for the next track. All of these activities require significant automation and robust execution including closed loop control, retries, and contingency handling.

As part of the DS-T task, the AI group deployed a planning system to automate the generation of control scripts. In order to provide automated operations capability, the DS-T station employs tightly coupled state-of-the-art hardware and software. Because of the complexity of operation of this equipment, the DS-T script generator is needed to custom tailor the control script for the specific combination of tracking goals (e.g., downlink, uplink, ranging) (Estlin et al., 1999). What makes this problem difficult is that many tasks must be sequenced in order to command the station. Furthermore, depending on the exact combination of services (or track goals) requested, different tasks may be required. Additionally, many of these tasks interact depending on the overall set selected.

The DS-T script generator (SG) uses artificial intelligence planning techniques to perform a complex software module reconfiguration process (Chien et al., 1998). This process consists of piecing together numerous highly interdependent smaller control scripts in order to produce a single script to control the operations of the DS-T station. The general software module reconfiguration problem encompasses a wide range of problems including the two science data analysis problems described in the previous section.

The core engine used in the SG is the automated scheduling and planning environment (ASPEN) (Fukanaga et al., 1997). The ASPEN system is a reusable, configurable, generic planning/scheduling application framework that can be tailored to specific domains to create plans/schedules. It has a number of useful features including an expressive modeling language, a constraint management system for representing and maintaining antenna operability and/or resource constraints, a temporal reasoning system, and a graphical interface for visualizing plans and states. For this domain, ASPEN has been tailored to accept input antenna tracking goals and then produce

the required command sequence necessary to create the requested link.

The control script produced by the SG sets up the track by performing the following steps:

1. Configure the station during pretrack.
2. Provide the track service requested by commanding the antenna and subsystems to acquire and maintain lock on the signal throughout mode changes.
3. Cleanup and shutdown the station at the completion of the track.

The DS-T concept was validated through a number of demonstrations. The demonstrations began with the automation of partial tracks in April 1998, continued with 1-day unattended operations in May, and concluded with a 6-day autonomous "*lights-out*" demonstration in September 1998. Throughout these demonstrations ASPEN was used to automatically generate the necessary command sequences for a series of Mars global surveyor (MGS) downlink tracks using the equipment configuration at Deep Space Station 26 (DSS26), a 34-meter antenna located in Goldstone, CA. These command sequences were produced and executed in a fully autonomous fashion with no human intervention. During the September demonstration, DS-T performed all Mars Global Surveyor coverage scheduled for the Goldstone antenna complex. This corresponded to roughly 13 hours of continuous track coverage per day.

As a component of the DS-T demonstrations, the SG performed flawlessly, producing dynamically instantiated control scripts based on the desired service goals for the communications pass as specified in the service request. The use of such technology resulted in three primary benefits:

- Autonomous operations enabled by eliminating the need for hundreds of manual inputs in the form of control directives. Currently, the task of creating the communications link is a manual and time-consuming process which requires operator input of approximately 700 control directives and the constant monitoring of several dozen displays to determine the exact execution status of the system.
- Reduced the level of expertise required of an operator to perform a communication track. Currently, the complex process requires a high level of expertise from the operator, but through the development of the KB by a domain expert this expertise is captured within the system itself.
- The KB provides a declarative representation of operation procedures. Through the capture of this expertise the KB documents the procedural steps of performing antenna communication services.

Spacecraft Commanding

Generating command sequences for spacecraft operations can be a laborious process requiring a great deal of specialized knowledge. Typically, spacecraft command sets are large, with each command performing a low-level task. There are often many interactions between the commands relating to the state of the spacecraft. In addition, due to spacecraft power and weight limitations, the resources available onboard spacecraft are often scarce. These factors in combination make manual generation of command sequences a difficult process. Because of the importance and expense of this process, tools to assist in planning and scheduling spacecraft activities are critical to reducing the effort (and hence cost) of mission operations.

The artificial intelligence group has an ongoing effort in developing and deploying automated planning and scheduling technology for spacecraft commanding. In 1997, a general system was deployed that uses artificial intelligence planning and scheduling technology, which was used to automatically generate command sequences for the DATA-CHASER shuttle payload operations. The DATA-CHASER automated planner/scheduler (DCAPS) architecture presented supports direct, interactive commanding, rescheduling and repair, resource allocation, and constraint maintenance (Chien et al., 1999b).

The DATA-CHASER automated planner scheduler implements search algorithms for two problems: initial schedule generation and schedule repair/refinement. In initial schedule generation, DCAPS generates a default schedule to perform science observations from the null schedule (i.e., an empty schedule). DATA-CHASER automated planner scheduler supports domain specific and randomized initial schedule generation strategies. In schedule repair/refinement, DCAPS accepts an existing schedule with conflicts (i.e., resource over subscription, state conflicts, etc.) and performs operations to make the schedule consistent with the spacecraft constraints. DATA-CHASER automated planner scheduler implements this functionality by using "iterative repair" search techniques (e.g., Zweben et al., 1994). Basically, this technique iteratively selects a schedule conflict and performs some action in an attempt to resolve the conflict. In iterative repair mode, DCAPS is naturally well adapted for human interaction. In this mode, a user can move, add, and delete activities in order to alter the schedule to their preferences. DATA-CHASER automated planner scheduler can then be invoked to repair state, resource, and temporal constraints caused by these modifications. Using an automated planner/scheduler (e.g., DCAPS) in this fashion, command sequence generation can be performed by scientists who need not be spacecraft and sequence engineer experts. This allows the scientist to become directly involved in the command sequencing process. Additionally, if there are changes in the spacecraft state (e.g., faults) or user-

defined goals (e.g., science opportunities), the repair algorithm allows simple rescheduling that attempts to minimize disruption of the original schedule. Finally, the highly restrictive payload resources and constraints are constantly monitored and conflicts automatically avoided.

The DCAPS system was developed for operation of the DATA-CHASER shuttle payload, which was developed and managed by students and faculty of the University of Colorado at Boulder. DATA-CHASER is a science payload, with a primary focus on solar observation. The main activities for the payload involve science instrument observations, data storage, communication, and control of the power subsystem. Science is performed using three solar observing instruments: the far ultraviolet spectrometer (FARUS), soft X-ray and extreme ultraviolet experiment (SXEE), and Lyman-alpha solar imaging telescope (LASIT). These are imaging devices that operate at various spectra.

The payload resources include power, tape storage, local memory, the three instruments, and the communication bus. DATA-CHASER is also constrained by externally driven states such as the shuttle orientation and external events such as shuttle venting of waste materials, which affect when certain science activities can be scheduled. Payload activities must be sequenced while avoiding or resolving conflicts.

Carrying the DATA-CHASER payload, STS-85, the Space Shuttle Discovery launched 7:41 AM PST on Thursday August 7, 1997 (Figure 6 shows the DATA-CHASER payload in the shuttle bay). Mission operations, including mission planning and scheduling, were performed for the 2-week flight. During the first 5 days of DATA-CHASER operations, DCAPS was used in manual mode. In this mode, activities were placed manually and DCAPS was used to validate constraints, identify constraint violations, and generate the actual command files. During the last 7 days of the payload operation, DCAPS was used to automatically generate schedules. In this

FIGURE 6. DATA-CHASER payload in STS-85 shuttle bay.

phase the domain-specific initial schedule generator was used to generate an initial schedule. Due to network lag times (DATA-CHASER was operated primarily from Colorado Space Grant and due to machine shortages DCAPS was running at JPL) use of the iterative repair techniques were somewhat limited. However, this turned out not to impact operations significantly; in many cases minor conflicts were repaired manually.

The DCAPS automated scheduling capability significantly impacted DATA-CHASER mission operations. DATA-CHASER automated planner scheduler enabled an 80% reduction in the amount of effort to produce operations plans. Manual generation of a 6-hour operations plan would require from 30–60 minutes in manual mode of operations and from 7–9 minutes using the DCAPS automated scheduling capability. This reduction in effort is because DCAPS can automatically generate an acceptable or near acceptable schedule very quickly. The number of modifications (if any) to make a DCAPS generated schedule acceptable can be made far faster than manually generating a schedule from scratch. DATA-CHASER automated planner scheduler also enabled a 40% increase in science return. Manually generated plans had 2–3 instrument scans per viewing opportunity, whereas DCAPS-generated plans had 3–4 scans per viewing opportunity. This is because DCAPS could directly monitor and track all of the complex timing constraints involved in the instrument activities and pack activities more tightly than operators manually placing instrument activities. During this 7 days of DCAPS automated use, DCAPS scheduled a total of 93 science scans and 202 payload commands. Figure 7 shows a payload operator using DCAPS to command the shuttle payload.

One significant feature of the DCAPS system is its declarative representation of flight rules and spacecraft constraints. This feature was tested during the STS-85 flight in the following manner. When initial command sequences were uplinked, a number of commands immediately following a reset command were rejected by the flight software. This was due to the fact that the initial flight rules were constructed with the understanding that immediately following a reset, commands could be issued to the payload.

FIGURE 7. Payload operator Jason Willis uses DCAPS to command the DATA-CHASER payload.

Actual operations showed that a delay of 30 seconds was needed before the payload could accept commands. When this problem was noticed and isolated, it was a simple manner to quickly update the DCAPS model to require this delay so that future command sequences would execute without problems. This aspect of each of modification is key in that spacecraft characteristics and operating procedures constantly evolve throughout the mission lifecycle as the spacecraft characteristics and mission priorities evolve.

LESSONS LEARNED AND ISSUES REALIZED

In this section a number of challenges are described that have been encountered in attempting to deploy planning and scheduling systems in several domains, including science data analysis, ground station automation, and spacecraft commanding. Specifically, discussed are issues relating to HTN and operator-based representations for planning, representing and reasoning about plan quality, acquiring, validating, and maintaining planning knowledge bases, and integration of planning and execution.

HTN Versus Operator Planning

Applied intelligence planning researchers have developed numerous approaches to the task of correct and efficient planning. Two main approaches to this task are *operator-based* planners and *hierarchical task network* (HTN) planners. In this section, a number of advantages and disadvantages of these approaches are described in light of experiences in developing several real-world planning systems (Estlin et al., 1997). While both HTN and operator-based planners typically construct plans by searching in a plan space, they differ considerably in how they express plan refinement operators. Hierarchical task network planners generally specify plan modifications in terms of flexible task reduction rules. Operator-based planners perform all reasoning at the lowest level of abstraction and provide strict semantics for defining operator definitions. By virtue of their representation, HTN planners more naturally represent hierarchy and modularity. In contract, operator-based plan refinements are more general since they can cover many more planning situations.

In the applications presented in this paper, a hybrid planning approach has been utilized that combines these two planning techniques. This approach has proven to be an effective method for planning in real-world applications. In developing these systems, the critical issue of planning representation has been specifically examined. If domain knowledge can be naturally represented in a planning system then

1. It will be easier to encode an initial knowledge base;

2. fewer encoding errors will occur, leading to a higher performance system;
3. maintenance of the knowledge base will be considerably easier.

Thus, an important measure for evaluating HTN and operator-based planning is how naturally each paradigm can represent key aspects of planning knowledge. Focus has been on four criteria: generality, hierarchy, flexibility, and efficiency.

Many of the obstacles in applying planning techniques to real-world problems can be characterized as representation difficulties. One advantage to employing an HTN planner is the ability to use abstract representation levels of domain objects and goals. This type of information is represented by constructing an object or goal hierarchy, where more detailed information such as object instances is at one end of the hierarchy, and more general information such as broad types is at the other end. For instance, in the DSN domain, different types of equipment (e.g., antennas, receivers) are represented in this format. Allowing abstract representations of these items enables one to represent the domain in an object-oriented form, which is easier to write and reason about. Decomposition rules can refer to either low- or high-level forms of a particular object or goal. Thus, more general rules about receivers can refer to a high-level receiver object, while more specialized information about particular receiver types is kept in smaller, more detailed rules which refer to the related low-level receiver type. This format allows domain information to be easily understood and updated since domain details are kept separate from more general knowledge. Overall, this representation contributes to a more concise and general domain knowledge base.

Unfortunately, a modular representation often makes it difficult to represent more specialized intermodular constraints. These types of constraints refer to information inside of several different decomposition rules and are usually only applicable in certain situations. For example, when performing receiver calibration in the DSN domain, it is sometimes necessary for high-level rules to refer to low-level steps in order to represent certain constraints as opposed to referring to a more general step. When using the Block-IV receiver, very long baseline interferometry (VLBI) telemetry tracks directly improve high-level ordering constraints on specific receiver calibration steps, instead of on a more general *calibrate-receiver* goal. Figure 8 shows the receiver calibration steps required for two different VLBI tracks; the left uses a Block-V receiver and the right a Block-IV receiver. The shaded portions of the graph indicate the specific steps that directly calibrate the receiver. In the Block-V case, receiver calibration is mapped onto a single general operator. However, in the Block-IV case it corresponds to five low-level steps, which have ordering constraints imposed on them by the telemetry rule. Most importantly, some of the ordering constraints from outside the receiver cali-

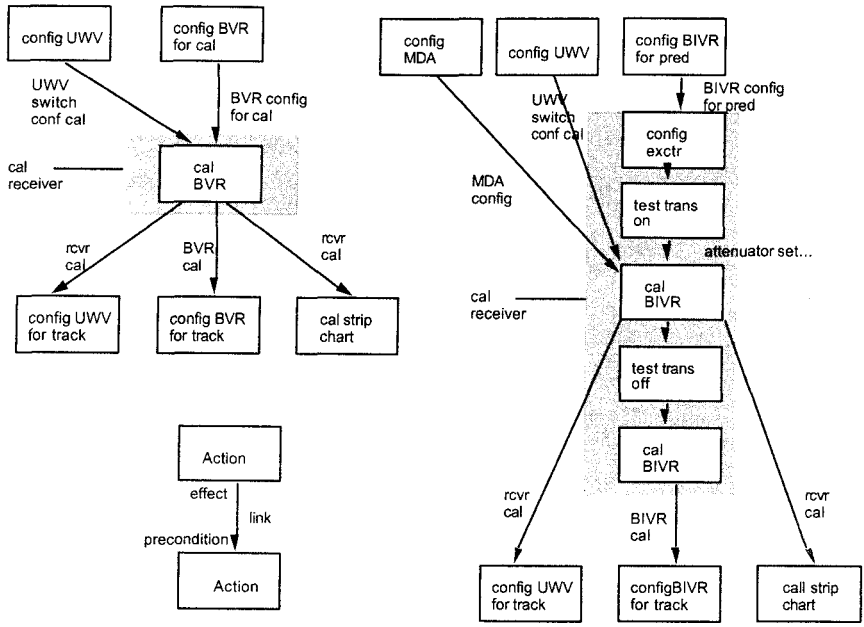

FIGURE 8. VLBI Receiver Subplans.

bration steps must refer to specific steps within the calibration subplan. The ordering constraints (4) refer to the middle step in the sublan (as opposed to being before all of the steps or after all of the steps). To encode this information in an HTN framework, one must use separate rules for tracks that require these intermodular constraints. Unfortunately, this solution results in less rule generality and increases the complexity of the domain definition. A more satisfactory solution is to incorporate operator-based planning techniques with the hierarchical representation. Instead of directly adding these constraints to decomposition rules, one can implicitly represent them by adding preconditions and effects to low-level track steps. This approach permits intermodular ordering constraints to be separate from decomposition rules, thereby allowing rules to retain their modularity. Thus, HTN approaches have the advantage of easily supporting a hierarchical representation. Operator-based approaches have the advantage of generality, since they can cover many planning situations unconsidered by the knowledge engineer. A hybrid HTN/operator-based approach allows an encoding that supports hierarchy and generality, without requiring an overly large search space.

Another advantage to using a hybrid system is the ability to encode implicit constraint information. These are constraints that may not be obvious when defining decomposition rules or operators, but are still necessary for correct planning. For instance, in the DSN domain, it is important

that an antenna not be moved to point at the final set of coordinates until after precalibration has taken place, due to stray transmissions that occur during some precalibration steps. One way to enforce this constraint is to explicitly add ordering constraints to all precalibration rules that will cause the antenna to be in a "stow " position. Unfortunately, such a constraint may have to be specified numerous times if there are multiple rules to which it applies. Another option is to use operator-based precondition/effect analysis. Unfortunately, this option requires a number of extra preconditions to be added and could possibly induce more search. A better solution is to utilize both HTN and operator-based techniques. First, one can add a protection to the main precalibration decomposition rule that forbids stray transmissions when the antenna is "on point" during the entire precalibration process. Then, using operator-based methods, one can use conditional effects to ensure that this requirement is enforced. Based on experience, hybrid planning methods offer the greatest flexibility in representing these types of implicit constraints.

Another notable difference between HTN and operator-based approaches is that the HTN approach allows the encoding of specific action sequences, while an operator-based approach often incurs significant search to construct this same sequence. Conversely, when operators can be combined in many different ways but still have interactions, an operator-based representation can be a more concise, natural method of encoding these constraints. In order to demonstrate this tradeoff an experiment was performed where a knowledge engineer (KE) encoded a simplified portion of the MVP image processing domain (Chien & Mortensen, 1996). This portion represented a subproblem called image navigation, which is one of the most complex subproblems in this domain. The KE developed three planning models, one in which only operator-based techniques are used, one where only HTN techniques were used, and one where both techniques were used. Based on the results of this experiment, the pure operator-based representation is inefficient from a search perspective. While only a small subset of the operator combinations will actually be used in solving problems, this type of framework requires that all operators be sufficiently accurate to rule out all other combinations. Representing this problem in a pure HTN framework was also difficult. Many complex combinations of dependencies and interrelations require numerous decomposition rules. In the combined HTN and operator-based framework, it is possible to represent different parts of the plan generation process using operator-based and/or HTN methods. Basic sequences can be easily represented using HTN rules. More complex additions to each basic sequence can be represented through operator-based constructs such as preconditions and condition effects. This approach resulted in a reduced number of rules (i.e., compactness) and avoidance of redundancy in the KB.

Plan Quality

One hurdle that must be faced when applying planning techniques to real-world problems is effectively representing and reasoning about plan quality (Chien et al., 1996). In this section, some of the challenges that have been encountered in this area for image processing are discussed—ground-station automation and spacecraft commanding domains.

In the MVP image-processing application, an important concern is output image quality. For a planning system to be able to represent large portions of an analyst's expertise, the planner must be able to represent and reason about the effect of various image transformations on image quality. For example, one of the most common image processing requests is for mosaicking, which is the process of combining a number of smaller images into a larger image. A frequent situation in mosaicking is that some of the images can be navigated absolutely—these images contain features that make it possible to exactly align these images (this is called absolute navigation). However, the remainder of the images can only be correctly placed on the ouput image by matching up points that are believed to be common between them and other images (tiepoints). This is a more difficult process known as relative navigation. When performing relative navigation, there are various measures of the confidence of the navigation process (such as residual errors). When processing these images, in order to produce a high quality image, the VICAR script must take into account the relative confidence values of alignment information from these different sources. Information from absolute navigation should be weighed more than information from relative navigation. And, relative navigation information is of varying degrees of confidence. Ideally, an expert image processing planning system would be able to reason about the navigation process and various measures of image quality, to determine at runtime the best order in which to process the images. In the best situation, the operator and plan representation would be able to explicitly represent these measures of image quality and at runtime execute the steps to ensure a high quality image. This would require characterizations of plan quality relating to measurable runtime attributes. A secondary, but also important concern for MVP, is the computational efficiency of the produced plan. If the image quality will be equivalent, there are sometimes different methods of achieving the same image processing goals but with different characteristics of computer runtime or disk storage. If MVP can reason about these types of costs for plans, it will be able to produce plans that are more acceptable to the analysts and scientists.

Plan quality is a key concern when planning for ground station automation. Producing high quality plans can greatly help reduce DSN operation costs. Important issues include generating efficient plans that can reduce

track setup and execution time and generating robust and/or flexible plans that facilitate both error recovery and last minute track adjustments. One important quality goal is to minimize the setup and execution time of a track plan. First, the time to setup (precalibration) and reset (post-calibration) the communications link should be minimized. Reducing this execution time allows more data to be returned per link setup. For instance, it can take up to 2 hours to manually precalibrate a DSN 70-meter antenna communications link for certain types of missions. Using an automated planning system, this time can be reduced to approximately 30 minutes, where further reductions in set-up time are limited by physical constraints of the antenna subsystems themselves. Changes in post-calibration can also reduce precalibration time for a subsequent track. For instance, if a following track requests a similar antenna operation to the one being currently executed, it may be unnecessary and wasteful to reset many of the antenna subsystems. Since many of the system settings will not vary between the two tracks, resetting these systems will only cause extra time to be spent on recalibration during the second track. These types of reductions in operations time can save thousands of dollars each time precalibration is performed. For this reason, plan execution time is a primary measure of plan quality.

Track execution time can often be significantly reduced by exploiting parallel path execution where the control of multiple subsystems is involved. When developing a planning system to automatically generate antenna-track plans, one would like the system to reason about plan execution time as a measure of plan quality. Since there can often be more than one correct plan for a particular antenna operation, it is important for a planning system to be able to compare a set of final plans using user identified plan quality measures. The planner, ASPEN, uses AI scheduling techniques to schedule nonconflicting track steps in parallel to help provide a shorter plan execution time. Other heuristics can also be easily incorporated that emphasize minimizing plan execution time.

Another important quality goal is to produce flexible plans that can check equipment status and respond accordingly (e.g., if the antenna is out of lock then reacquire the signal, or if wind speeds are too high then delay tracking). In particular, producing more flexible plans can allow for easier error recovery. For example, an antenna directive may fail and require new commands to be inserted into the plan, which may change some state conditions and time tags. The plan needs to be flexible enough so that later commands can still be properly executed even though these changes have occurred. Also, flexibility can allow for easier replanning. A planning system used for ground-station automation may have to replan during the course of typical antenna operations. More flexible plans will allow for more efficient replanning where plan modifications can be quickly made, such as inserting new steps into a plan at various points. Flexible plans that allow for modifi-

cation while still retaining their applicability are greatly valued since they allow execution to quickly continue. The need for replanning is discussed in more detail in the Integrated Planning and Execution section.

Plan quality is also a concern for spacecraft mission planning. Generating spacecraft command sequences require consideration of both hard and soft constraints. The planner must be able to find plans that satisfy all relevant hard constraints, which represent *necessary* constraint information, and that appropriately optimize over soft constraints, which represent more optional plan-quality improvements. Work in this area first concentrates on a study of the metaphors used by mission operations personnel to specify their preferences frequently occurring in missions operations. Current experience indicates that these preferences can be divided into three classes: 1) activity preferences, 2) state preferences, and 3) resource preferences. Activity preferences concern the existence and placement of activities in the plan. Many of these preferences relate to the temporal placement of an activity or to activity duration. Activity preferences may also include preferences to maximize the number of activities, such as observations. A second class of preferences involves state variables, which represents the state of a device or subsystem over time (e.g., camera is "on" or "off", aperture is "open" or "closed"). Examples of this type of preference include constraints such as keeping the imaging device closed when not in use or the preference to minimize the power cycling of an instrument. A third class of preferences concerns spacecraft resources (e.g., battery power, propellant). This class of preferences would include choosing to minimize propellant usage or choosing the thermal range of the spacecraft.

Once these preference language constructs have been identified, the next step is to develop a formal semantics that defines the objective function for plan optimization. This will enable a clear semantics for search and dominance of one plan over another. Once the preference language has been specified and the semantics formalized, a formalization of local and global optima is then developed based on these preference types. This work concentrates on developing characterizations of local optima for heuristic search strategies that will hopefully be able to enable quick discovery of good solutions with local optima guarantees (Aarts & Korst, 1990). In cases where it is necessary to find higher quality solutions (with corresponding higher search cost), random restart methods could be used to improve solution quality.

Knowledge Acquisition, Modeling, and Maintenance

A key bottleneck in applying AI planning techniques to a real-world problem is the amount of effort required to construct, debug, verify, and update (maintain) the planning knowledge base (Chien, 1998). In particular,

planning systems must be able to compare favorably in terms of software lifecycle costs to other means of automation such as scripts or rule-based expert systems. An important component to reducing such costs is to provide a good environment for developing planning knowledge bases. Despite this situation, relatively little effort has been devoted to developing an integrated set of tools to facilitate constructing, debugging, verifying, and updating specialized knowledge structures used by planning systems.

While considerable research has focused on knowledge acquisition systems for rule-based expert systems (Davis, 1979) and object-oriented/inheritance knowledge bases with procedures and methods (Gil & Tallis, 1995), little work has focused on knowledge acquisition for specialized planning representations. Notable exceptions to this statement are DesJardins (1994), which uses inductive learning capabilities and a simulator to refine planning operators and Wang (1995) which uses expert traces to learn and a simulator to refine planning operators. However, in many cases a simulation capability is not available. In these situations the user needs assistance in causally tracing errors and debugging from a single example. This assistance is sorely needed to enable domain experts to write and debug domain theories without relying on AI people. Furthermore, planning knowledge base maintenance is often overlooked. Such tools are also invaluable in tracking smaller bugs, verifying KB coverage, and updating the KB as the domain changes.[1] While these tools can draw much from causal tracking techniques used in rule-based system (Davis, 1979), there are several aspects of planning systems that differentiate them from rule-based systems—their specialized representations and temporal reasoning capabilities. Two specialized representations for planning are prevalent—task reduction rules and planning operators. These representations, as well as the most common constraints (ordering and codesignation constraints), have evolved so that specialized reasoning algorithms must be adapted to support debugging.

Many types of knowledge encoding errors can occur: incorrectly defined preconditions, incorrectly defined effects, and incorrect variable specifications. Invariably, the end result is a mismatch between the planners model of the legality of a plan and the model dictated by the domain (or domain expert). Thus, the end symptoms of a knowledge base error can be broadly classified into two categories.

Incorrect Plan Generation

This occurs when the planner is presented a problem and generates a plan that does not achieve the goals in the current problem context. By experience, the current problem and faulty solution can focus attention in debugging the flaw in the knowledge base. By using the faulty plan to direct the debugging process, the user can often focus on the incorrect link in the plan (faulty protection or achievement), allowing for rapid debugging.

Failure to Generate a Plan

This occurs when the planner is presented with a solvable problem, but the planner is unable to find a solution. By experience, this type of failure is far more difficult to debug. This is because the user does not have a particular plan to use to focus the debugging process. Thus, often a user would manually write down a valid plan based on their mental model of the domain, and then trace through the steps of the plan to verify that the plan could be constructed. Because experience has been that detecting and debugging failure-to-generate-a-plan cases has been more difficult, this work focuses on:

1. Verifying that a domain theory can solve all problems deemed solvable by the expert;
2. Facilitating debugging of cases where the domain theory does not allow solution of a problem deemed solvable by the domain expert.

In general, most work on debugging knowledge bases has presumed the existence of a sound and complete performance element (in this case the planner), e.g., the problem-solving method and knowledge base are correct. In theory these properties are also presumed but in practice it is acknowledged that the planner is likely to be sound but incomplete (due to the size of the search space and the general difficulty of proving a problem unsolvable). However, practically speaking, the knowledge engineer is interested in developing a knowledge base that a specific planner can use to solve the problems of interest, rather than developing a knowledge base that in theory specifies that a solution exists. Thus, tautologically the planner can be viewed as complete, and the goal is to develop a knowledge base that is semantically correct and organized such that the planner can produce solutions for the relevant problems.

In response to the difficulties in developing and maintaining planning knowledge bases, two types of tools have been developed to assist in developing planning knowledge bases—static analysis tools and completion analysis tools (Chien 1998). Static analysis tools analyze the domain knowledge rules and operators to see if certain goals can or cannot be inferred. However, because of computational tractability issues, these checks must be limited. Static analysis tools are useful in detecting situations in which a faulty knowledge base causes a top-level goal or operator precondition to be unachievable—frequently due to omission of an operator effect or a typographical error.

Completion analysis tools operate at planning time and allow the planner to complete plans that can achieve all but a few focused subgoals or top-level goals. Completion analysis tools are useful in cases where a faulty knowledge base does not allow a plan to be constructed for a problem that

the domain expert believes is solvable. In the case where the completion analysis tool allows a plan to be formed by assuming goals true, the domain expert can then be focused on these goals as preventing the plan from being generated.

In general, the classical planning task is undecidable (Chapman, 1987). Because the static and completion analysis techniques in general operate by solving planning problems, they too are solving undecidable problems. Of course, the propositional versions of the algorithms later presented are exceptions. In practice, of greater importance than the computational class of these algorithms is their performance on the actual problems encountered. These issues are briefly discussed in the section on Evaluation and Use.

In addition, in the area of spacecraft commanding, the assistance of spacecraft operations personnel has been enlisted in developing the ASPEN modeling language (Sherwood et al., 1998, Smith et al., 1998). Automated scheduling and planning environment uses constructs based on common spacecraft operations metaphors, facilitating the operations modeling effort and enabling non-AI personnel to construct and debug models.

Integrated Planning and Execution

Historically, planning research has focused on a batch formulation of the problem. In this approach planning is viewed as a step in the "sense, plan, execute" paradigm in which plans are generated from scratch in an off-line fashion. With automated planning being used in a wider range of applications, it is becoming increasingly apparent that plans are typically generated once and revised numerous times in a continuous fashion. In this section these shortcomings are illustrated in the context of spacecraft operation and ground station planning scenarios.

An autonomous real-time control system must balance long-term and short-term considerations. It must perform purposeful activities that ensure long-term science goals, and short-term operations goals are achieved and ensure that it maintains positive resource margins. This requires planning in advance to avoid a series of shortsighted decisions that can lead to failure. However, it must also respond in a timely fashion to a somewhat dynamic and unpredictable environment. In terms of high-level, goal-oriented activity, command sequence plans must often be modified in the event of fortuitous events such as events completed early and setbacks such as failure to acquire a guidestar for a science observation. Here is briefly described an integrated planning and execution architecture that supports continuous modification and updating of a current working plan in light of changing operating context.

As already mentioned, an autonomous control system must respond in a timely fashion to a (somewhat) dynamic, unpredictable environment. This

situation is called *dynamic planning*, in which a plan must be continually updated in light of changing operating context. In such an operations mode, a planner would accept activity and state updates through a real-time interface. Making the planner more timely in its responses has a number of benefits:

- The planner can be more responsive to unexpected (e.g., unmodelable) changes in the environment that would manifest themselves as updates on the execution status of activities as well as monitored state and resource values.
- The planner can reduce reliance on predictive models (e.g., inevitable modeling errors), since it will be updating its plans continually.
- Fault protection and execution layers need worry about controlling over a shorter time horizon (as the planner will replan within a shorter time span).
- Because of the hierarchical reasoning taking place in the architecture, there is no hard distinction between planning and execution—rather more deliberative (planner) functions reside in the longer-term reasoning horizons and the more reactive execution functions reside in the short-term reasoning horizons. Thus, there is no planner to executive translation process.

In a traditional "plan, sense, act" cycle, planning is considered a batch process and the system operates on a relatively long-term planning horizon, not making the system very responsive to change. To achieve a higher level of responsiveness, a *continuous planning* approach is utilized (Chien et al., 1999a). Rather than considering planning a batch process in which a planner is presented with goals and an initial state, the planner has a current goal set, a current state and projections into the future, and a *current plan.* At any time an incremental update to the goals or current state may update the planner process. This update may be an unexpected event or simply time progressing forward. The planner is then responsible for maintaining a consistent, satisfying plan with the most current information. This current plan and projection is the planner's estimation as to what it expects to happen in the world if things go as expected. However, as things rarely go exactly as expected, the planner stands ready to continually modify the plan. Current iterative repair planning techniques enable incremental changes to the goals, initial state, or plan, and then iteratively resolve any conflicts in the plan. After each update, its effects will be propagated through the current projections, conflicts identified, and the plan updated (e.g., plan repair algorithms invoked).

In this approach, the real-time software produces updates that require responses by near and long-term activities for the system. The system's state

is modeled by a set of timelines, which represent the current and expected evolution of the system over time. This model includes the current state and the projection of how the state will evolve in light of actions expected to take place in the future. These actions are the current plan that is also reflected in the timelines as actions at future points in time.

At each iteration through the loop, as the world changes, the actual state of the system drifts from the state expected by the timelines. The real-time software updates the timeline models with notifications of actual state values, resource values, start times, and completion times for activities. Each of these updates, when synchronized with the current plan, may introduce conflicts. A conflict is when an action in the plan is inappropriate—because its required state and/or resource values violate the system constraints. Whenever such a conflict exists the planner notes the conflict and performs plan modifications to bring the plan back into sync with the current state and future projections. Because this process is continuous, the plan rarely has the opportunity to get significantly out of sync. As a result the high-level actions of the system are more responsive to the actual state.

The continuous planner approach is being developed in the context of two different systems: one for ground station automation and commanding (closed loop execution and recovery (CLEaR)), and another for spacecraft commanding as part of a larger common mission data systems (MDS) task. For the MDS task a prototype was developed to demonstrate the feasibility of a continuous planning approach to spacecraft commanding. The prototype was demonstrated by running on operational scenarios from the Deep Space 4/Champollion (DS4) mission. This is a mission scheduled to launch in 2003 in which a spacecraft lands on a comet in 2007 and returns a sample of the comet to the earth in 2010.

In the simulation, the planner must take several soil samples and perform in-situ science on these samples. The planner must coordinate drilling and collecting of three soil samples and perform experiments (which include baking two of the samples in an oven to study the gases released during the heating) and storage of the third sample for return to earth.

In this demonstration three different class of events were demonstrated:

1. An error in the oven was introduced, which resulted in the continuous planner (CP) moving the remaining activities to the backup oven and continuing with operations.
2. The model of the estimated data compression was inaccurate causing the data buffer to unexpectedly fill. In response, the planner was able to foresee a data volume difficulty and respond by adding an uplink activity to the schedule to free space for the remaining experiments.
3. The last capability demonstrated again resulted in the lack of accuracy in the model. In this case, the simulator unexpectedly decreases the amount

of power left in the battery. The planner projects that this does not leave sufficient energy to complete the experiments and liftoff to return the sample. Recognizing that this is a critical part of the mission the planner removes the final experiment from the schedule to ensure adequate power to complete the mission.

In conjunction with this incremental, continuous planner approach, a hierarchical approach to planning is also being advocated. In this approach, the long-term planning horizon is planned only at a very abstract level. Shorter and shorter planning horizons are planned in greater detail, until finally at the most specific level the planner plans only a short time in advance (just in time planning).

The idea behind this hierarchical approach is that only very abstract projections can be made over the long term and that detailed projections can only be made in the short term. Hence, there is little utility in constructing a detailed plan far into the future—chances are it will end up being replanned anyway. At one extreme the short-term plan may not be "planned" at all and may be a set of reactions to the current state in the context of the near-term plan. This approach can be implemented in a control loop by making high-level goals active regardless of their temporal placement, but medium and low-level goals are only active if they occur in the near future. Likewise, conflicts are only regarded as important if they are high-level conflicts or if they occur in the near future. As the time of a conflict or goal approach, it will eventually become active and the elaboration/planning process will then be applied to resolve the problem.

CONCLUSION

A number of lessons learned have been described in deploying automated planning and scheduling systems for space applications at JPL. Specifically, described have been issues relating to: HTN and operator-based representations for planning, representing, and reasoning about plan quality; acquiring, validating, and maintaining planning knowledge bases; and integration of planning and execution. These issues have been described and discussed in the context of systems that have been fielded in the areas of science data analysis, ground station automation, and spacecraft autonomy.

ACKNOWLEDGMENTS

This work was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration. Portions of this work were supported by the Autonomy Technology Program, managed by Richard Doyle at JPL, and with

Melvin Montemerlo as the headquarters program executive, NASA Code SM. Other portions of this work were supported by the Telecommunications and Mission Operations Technology Program, managed by Chad Edwards.

NOTES

1. For work in verifying rule-based systems, see O'Keefe and O'Leary (1993). For work on rule base refinement using training examples (the analogue of a simulator for planning KB refinement) see Ginsberg et al. (1988).

REFERENCES

Aarts, E., and J. Korst. 1990. *Simulated annealing and Boltzmann machines: A stochastic approach to combinatorial optimization and neural computing.* New York: John Wiley.

Chapman, D. 1987. Planning for conjunctive goals. *Artificial Intelligence* 32:3.

Chien, S. A. 1994. Using AI planning techniques to automatically generate image processing procedures: A preliminary report. In *Proceedings of the Second International Conference on AI Planning Systems,* Chicago, IL, 219–224.

Chien, S., R. Hill Jr., X. Wang, T. Estlin, K. Fayyad, H. Mortensen. 1996. Why real-world planning is difficult. A tale of two applications. In *Advances in planning,* ed. M. Ghallab. Washington, D.C.: IOS Press.

Chien, S. A., and H. B. Mortensen. 1996. Automating image processing for scientific data analysis of a large image database. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 18(8):854–859.

Chien, S., D. DeCoste, R. Doyle, and P. Stolorz. 1997a. Making an impact: Artificial intelligence at the jet propulsion laboratory. *AI Magazine* 18(1):103–122.

Chien, S., F. Fisher, E. Lo, H. Mortensen, R. Greeley. 1997b. Using artificial intelligence planning to automate science data analysis for large image databases. In *Proceedings of the 1997 Conference on Knowledge Discovery and Data Mining,* Newport Beach, CA.

Chien, S., F. Fisher, H. Mortensen, E. Lo, R. Greeley, A. Govindjee, T. Estlin, and X. Wang. 1998. Using artificial intelligence planning techniques to automatically reconfigure software modules. In *Tasks and methods in applied artificial Intelligence,* Lecture Notes in Artificial Intelligence, 1416, eds. A. Pasqual del Pobil, J. Mira, and M. Ali, 415–426, New York: Springer-Verlag.

Chien, S. 1998. Static and completion analysis for knowledge acquisition, validation, and maintenance of planning knowledge bases. *International Journal of Human Computer Studies* 48:499–519.

Chien, S., R. Knight, A. Stechert, G. Rabideau, and R. Sherwood. 1999a. A prototype for automated spacecraft planning and execution. In *Proceedings of the IEEE Aerospace Conference.* Aspen, CO.

Chien, S., G. Rabideau, J. Willis, and T. Mann. 1999b. Automating planning and scheduling of shuttle payload operations. *Artificial Intelligence Journal* 114:239–225.

Davis, R. 1979. Interactive transfer of expertise: Acquisition of new inference rules. In *Artificial Intelligence* 12(2): 121–157.

DesJardins, M. 1994. Knowledge development methods for planning systems. In *Working Notes of the AAAI Fall Symposium on Learning and Planning: On to Real Applications,* New Orleans, LA, 34–40.

Estlin, T., S. Chien, and X. Wang. 1997. An argument for an integrated hierarchical task network and operator-based approach to planning. In *Recent advances in AI planning,* eds. S. Steel and R. Alami, Lecture Notes in Artificial Intelligence. New York: Springer-Verlag, 182–194.

Estlin, T., F. Fisher, D. Mutz, S. Chien. 1999. Automated planning for a deep space communication station. In *Proceedings of the IEEE Conference on Robotics and Automation (ICRA),* Detroit, Michigan, May 1999.

Fisher F., E. Lo. S. Chien and R. Greeley. 1997. Using Artificial Intelligence planning to automate SAR processing for planetary geology. *NASA Science Information Systems,* Issue 42.

Fisher, F., S. Chien, L. Paal, E. Law, N. Golshan, and M. Stockett. 1998a. An automated deep space communications station. In *Proceedings of the 1998 IEEE Aerospace Conference*, Aspen, CO.

Fisher F., S. Chien, E. Lo, and R. Greeley. 1998b. Using Artificial Intelligence planning to automate SAR image processing for scientific data analysis. In *Proceedings of the 1988 Conference on Innovative Application of artificial intelligence*, Madision, WI.

Fisher, F., T. Estlin, D. Mutz, L. Paal, E. Law, M. Stockett, N. Golshan, and S. Chien. 1999. The past, present, and future of ground station automation within the DSN. In *Proceedings of the 1999 IEEE Aerospace Conference*, Aspen, CO.

Fukanaga, A., G. Rabideau, S. Chien, and D. Yan. 1997. Toward an application framework for automated planning and scheduling. In *Proceedings of the 1997 International Symposium of Artificial Intelligence, Robotics and Automation for Space*, Tokyo, Japan.

Gil, Y., and M. Tallis. 1995. Transaction-based knowledge acquisition: Complex modifications made easier. In *Proceedings of the 9th knowledge acquisition for Knowledge-based Systems Workshop*, Banff, Canada.

Ginsberg, A., S. Weiss, P. Politakis. 1988. Automatic knowledge based refinement for classification systems. *Artificial Intelligence* 35:197–226.

O'Keefe, R., and D. O'Leary. 1993. Expert system verification and validation: A survey and tutorial. *AI Review* 7:3–42.

Pree, W. 1995. *Design patterns for object oriented software development*. New York: Addison Wesley.

Sherwood, R., A. Govindjee, D. Yan, G. Rabideau, S. Chien, and A. Fukunaga. 1998. Using ASPEN to automate EO activity planning. In *Proceedings of the 1998 IEEE Aerospace Conference*, Aspen, CO.

Smith, B., K. Rajan and N. Muscettola. 1997. Knowledge acquisition for the onboard planner of an autonomous spacecraft. In *Advances in knowledge acquisition*, Lecture Notes in Artificial Intelligence (Proceedings EKAW-97). New York: Springer-Verlag.

Smith B., R. Sherwood, A. Govindjee, D. Yan, G. Rabideau, S. Chien, and A. Fukunaga. 1998. Representing spacecraft mission planning knowledge in Aspen. *AIPS-98 Workshop on Knowledge Engineering and Acquisition for Planning*. AAAI Technical Report WS-98-03.

Wang, X. 1995. Learning by observation and practice: An incremental approach for planning operator acquisition. In *Proceeding of the 1995 International Conference on Machine Learning*. Tahoe City, CA.

Zweben, M. B. Daun, E. Davis, and M. Deale. 1994. Scheduling and rescheduling with iterative repair. In *Intelligent scheduling*. San Francisco: Morgan Kaufman.